IN THE CLAIMS

Please cancel claims 1-9 without prejudice.

Please add the following new claims:

1-9. Cancelled.

10. A method of eliminating partial redundancy comprising:

(A) speculatively computing down-safety by ignoring rarely taken branches in a control-flow graph; and

(B) computing up-safety using the results of the down-safety calculation to determine where operations are speculatively available.

11. The method of claim 10, wherein the computation of down safety further comprises a lattice that distinguishes strict down-safety from speculative down-safety.

12. The method of claim 10, further comprising determining whether a routine includes cleanup instructions prior to computing the down-safety.

13. The method of claim 12, further comprising:

removing edges from an exceptional path if the routine does not include cleanup instructions; and

inserting a prologue at the beginning of the routine.

14. The method of claim 12, further comprising building a cleanup tree and cleanup states for the routine.

15. The method of claim 14, wherein the cleanup tree and cleanup states represents an exception handling (EH) stack at points where a exception may be thrown.

16. The method of claim 15, wherein the cleanup tree and cleanup states determine operations to be inserted.

17. The method of claim 10, wherein computing down-safety comprises:

computing a down-safety transfer function for each of a plurality of edges of an exceptional path; and

executing flow equations for the down-safety transfer functions.

18. The method of claim 17, wherein computing up-safety comprises:

computing an up-safety transfer function for each of a plurality of edges of an exceptional path; and

executing flow equations for the up-safety transfer functions.

19. The method of claim 18, further comprising inserting instructions to set components of an exception handling (EH) stack.

20. The method of claim 19, further comprising:

removing edges from the exceptional path; and

inserting a prologue at the beginning of the routine.

21 A method comprising:

(A) representing the program with a control-flow graph in which actions to take in event of an exceptional situation are represented by explicit paths in the graph;

(B) analyzing said program to determine at which points the stack must be in a valid state;

(C) building a forest of tress that represent the stack state at said points where:

(i) each node of the tree represents a possible item on the stack, and

(ii) a stack state is represented as a path from a tree node to the root of the tree;

(D) computing where to place operations that set the state of items on the stack by performing the steps of:

(i) constructing flow equations for down-safety of operations that set the state of items on the stack, where the equations ignore rarely taken branches,

(ii) solving said flow equations for down-safety of operations that set the state of items on the stack,

(iii) constructing flow equations for up-safety of operations that set the state of items on the stack, where the equations use the solution for down-safety to determine which setting operations are speculatively available, and

(iv) solving said flow equations for up-safety of operations that set the state of items on the stack,

(E) using the results of said flow equations to place instructions that maintain the stack;

(F) removing edges from the control-flow graph which represent said actions for exceptional events; and

(G) inserting a prologue at entry to the control-flow graph that saves the existing pointer to the top of the EH stack.